A Context Free Gramma for Key Noun-Phrase Extraction from Text

Ying Liu
St. John's University
liuy1@stjohns.edu

Abstract

Topic extraction is a major field in text mining. Key noun-phrases play a very important role in identifying the important document topic because the primary information of a document is described in noun-phrases. In this paper, we propose a new topic extraction schema to identify the key noun-phrases by constructing a context free grammar (CFG) from input documents. In our new method, documents are reconstructed as a set of CFG rules using an existing algorithm called Sequitur. The Sequitur algorithm infers the resulting context-free grammatical rules, which can be considered as a hierarchical structure, from a sequence of discrete symbols. The resulting hierarchical structure exposes the underlying structure of input sequence that can help us capture meaningful regularity. Based on this hierarchical structure of the input document, we designed a new algorithm to identify noun-phrases and extract key noun-phrases.

1. Introduction

The technique that identifies the concept of a document set is called Topic Extraction. Topic concept is important document knowledge that gives a high-level, concise and compact description of a document or a set of documents. It can sufficiently help reader understand the main contents included in the source documents. The topic can be represented as key phrases, which can be either single keywords or multiword key terms, or text summaries, which are sentences that describe the content of a group of documents. Key noun-phrases play a very important role in identifying the important document topic because the primary information of a document is described in noun-phrases and most concept terms are noun-phrases.

Because key phrases are sufficiently informative, they can also be used in various applications such as text clustering and classification [1, 2], thesaurus construction [3], document similarity analysis [4, 5], and retrieval engines [6, 7], etc. For example, key phrases can be used as a low-cost similarity measurement between documents and further cluster documents into groups based on the similarity.

The topic of a single document can be manually provided by authors in forms of key phrase assignment and abstract. However, it is a very laborious task to manually read through large numbers of documents and give an essential summary. Several effective techniques for automatic key phrase extraction have been developed [8, 9, 10, 11]. Some of these methods are machine learning algorithms that require training data to train their programs. Some methods rely on the special structural features in the documents. For example, KIP [12] requires a glossary database containing pre-identified key words in order to identify the key phrases.

In this paper, we propose a new topic extraction schema to identify the key noun-phrases by constructing a context free grammar (CFG) from input documents. The CFG provides a simple and precise mechanism for describing the document by which some phrases within the document are built from smaller blocks, capturing the "block structure" of sentences in a natural way. In our new method, first of all, the documents are reconstructed as a set of context-free grammar rules using an existing algorithm called Sequitur [13]. The Sequitur algorithm infers the resulting context-free grammatical rules, which can be considered as a hierarchical structure, from a sequence of discrete symbols. The basic insight is that phrases which appear more than once can be replaced by a grammatical rule that generates the phrase, and this process can be continued recursively, producing a hierarchical representation of the original sequence. The resulting hierarchical structure exposes the underlying structure of input sequence that can help us capture meaningful regularity. Based on this hierarchical structure of the input document, we designed a new algorithm to identify noun-phrases and extract key noun-phrases.

2. The Sequitur Algorithm

The Sequitur algorithm is a linear-time online algorithm [13] that forms a context-free grammar for a given string input. This algorithm is initially used in data compression software applications. Here is a brief

URI: https://hdl.handle.net/10125/59557
ISBN: 978-0-9981331-2-6

review of the algorithm. Starting with a rule with the non-terminal symbol *S* at the left hand side, the algorithm continuously fetches symbols from the input and appends them to the right hand side of the starting rule. Duplicate subsequences are checked during processing and a new production rule is generated for each repeated subsequence if this is not done before. After the new rule is generated, the repeated subsequence is replaced by the left hand side non-terminal symbol of the rule.

Two properties are ensured in the compressed grammar representation [13].

- *Diagram uniqueness: no pair of adjacent symbols appears more than once in the grammar.* If, by adding a new input symbol, two adjacent symbols appear more than once in the grammar, a new produce rule will be created to replace both appearances.
- *Rule utility: every rule is used more than once.* The number of times that a rule is used can decrease during the processing. If this number is reduced to one, the rule will be discarded.

To illustrate the Sequitur algorithm, an example similar to the one shown in [13] is provided in Table 1, which shows the grammars that result when successive symbols of the sequence *abcbabcbabcabc* are processed. The second column shows the sequence observed so far, the third column gives the grammar created from the sequence, and the fourth column notes constraints that have been violated, and actions that are taken to resolve the violations.

In the example shown in Table 1, after we add the fifth input symbol, we have the starting rule as follows:

$$S \rightarrow abcba$$

After adding the sixth symbol "b", we get "abcbab". Since the subsequence "ab" appears twice, it has to be replaced by a new rule. Sequitur creates the new rule *Z*, with *ab* as its right-hand side, and replaces the two occurrences of *ab* by Z. We then have:

$$S \rightarrow ZcbZ$$
$$Z \rightarrow ab$$

This illustrates the basic procedure for dealing with duplicate diagrams. Not every repeated diagram gives rise to a new rule. If there is an existing rule has a right-hand side of the new diagram, then no new rule need to be created. The non-terminal symbol of this existing rule replaces the repeated diagram. For example, before the 14th symbol "c" is added, we have:

$$S \rightarrow XXYZ$$
$$X \rightarrow Yb$$
$$Y \rightarrow Zc$$
$$Z \rightarrow ab$$

When the 14th symbol "c" is added in Table 1, *Zc* appears twice in the grammar, an existing rule *Y*, with *Zc* as its right-hand side, replaces the *Zc*. That is:

$$S \rightarrow XXYY$$
$$X \rightarrow Yb$$
$$Y \rightarrow Zc$$
$$Z \rightarrow ab$$

However, replacing the *Zc* leaves only one appearance of rule *Z*, violating the constraint of rule utility. For this reason, *Z* is removed from the grammar, and its right-hand side is substituted in the one place where it occurs. We have:

$$S \rightarrow XXYY$$
$$X \rightarrow Yb$$
$$Y \rightarrow abc$$

Rule *Y* now contains three symbols. This is the mechanism for forming long rules: form a short rule temporarily, and if subsequent symbols continue the match, allow a new rule to supersede the shorter one and delete the latter.

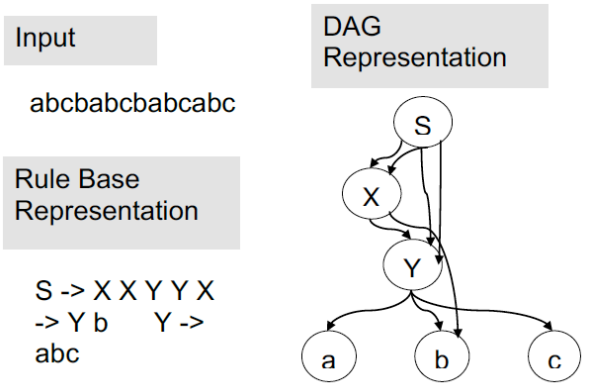

Figure 1 An example of the Sequitur algorithm

In summary, Sequitur's operation consists of ensuring both properties. When describing the algorithm, the properties act as constraints. The algorithm operates by enforcing the constraints on a grammar: when the diagram uniqueness constraint is violated, a new rule is formed, and when the rule utility constraint is violated, the useless rule is deleted. The result of the Sequitur algorithm is a set of context-free grammar rules. Alternatively, the result can be represented by a directed acyclic graph (DAG) as shown in Figure 1. The original input sequence can be reconstructed by traversing the first rule and recursively replace the non-terminal symbol with the right-hand side of the rule corresponding to the non-terminal symbol.

Table 1. An example of the Sequitur algorithm

Symbol number	The sequence so far	resulting grammar	remarks
1	a	$S \rightarrow a$	
2	ab	$S \rightarrow ab$	
3	abc	$S \rightarrow abc$	
4	abcb	$S \rightarrow abcb$	
5	abcba	$S \rightarrow abcba$	
6	abcbab	$S \rightarrow abcbab$	ab appears twice
		$S \rightarrow ZcbZ$ $Z \rightarrow ab$	enforce diagram uniqueness
7	abcbabc	$S \rightarrow ZcbZc$	Zc appers twice
		$S \rightarrow YbY$ $Y \rightarrow Zc$ $Z \rightarrow ab$	enforce diagram uniqueness Z is only used once
		$S \rightarrow YbY$ $Y \rightarrow abc$	enforce rule utility
8	abcbabcb	$S \rightarrow YbYb$ $Y \rightarrow abc$	Yb appears twice
		$S \rightarrow XX$ $Y \rightarrow abc$ $X \rightarrow Yb$	enforce diagram uniqueness. X is only used once
		$S \rightarrow XX$ $X \rightarrow abcb$	enforce rule utility
9	abcbabcba	$S \rightarrow XXa$ $X \rightarrow abcb$	
10	Abcbabcbab	$S \rightarrow XXab$ $X \rightarrow abcb$	ab appears twice
		$S \rightarrow XXZ$ $X \rightarrow Zcb$ $Z \rightarrow ab$	enforce diagram uniqueness X is only used once
11	abcbabcbabc	$S \rightarrow XXZc$ $X \rightarrow Zcb$ $Z \rightarrow ab$	Zc appears twice
		$S \rightarrow XXY$ $X \rightarrow Yb$ $Z \rightarrow ab$ $Y \rightarrow Zc$	enforce diagram uniqueness Z is only used once
12	abcbabcbabca	$S \rightarrow XXYa$ $X \rightarrow Yb$ $Y \rightarrow abc$	
13	abcbabcbabcab	$S \rightarrow XXYab$ $X \rightarrow Yb$ $Y \rightarrow abc$	ab appears twice
		$S \rightarrow XXYZ$ $X \rightarrow Yb$ $Y \rightarrow Zc$ $Z \rightarrow ab$	enforce diagram uniqueness Y is only used once
14	abcbabcbabcabc	$S \rightarrow XXYZc$ $X \rightarrow Yb$ $Y \rightarrow Zc$ $Z \rightarrow ab$	Zc appears twice
		$S \rightarrow XYYY$ $X \rightarrow Yb$ $Y \rightarrow Zc$ $Z \rightarrow ab$	enforce diagram uniqueness Z is only used once
		$S \rightarrow XYYY$ $X \rightarrow Yb$ $Y \rightarrow abc$	enforce rule utility

Table 2 summarizes the algorithm, which is also provided in [13]. Line 1 deal with new observations in the sequence. Lines 2 through 6 enforce the diagram utility constraint. Line 3 determines whether the new diagram matches an existing rule, or whether a new rule is necessary. Lines 7 and 8 enforce rule utility. Lines 2 and 7 are triggered whenever the constraints are violated.

Table 2 The Sequitur algorithm

```
1 As each new input symbol is observed, append it to rule S.

2 Whenever a duplicate diagram appears,
3   If the other occurrence is a complete rule,
4     Replace the new diagram with the non-terminal that heads the other diagram,
5   Otherwise
6     Form a new rule and replace both diagrams with the new non-terminal

7 Whenever a rule is used only once,
8   remove the rule, substituting its contents in place of the non-terminal
```

The Sequitur algorithm has been evaluated in data compression and found that it is competitive with the best compression algorithms, particularly when a large amount of text and constructive text is available [13]. But there is another important property: Sequitur represents a sequence as a hierarchical structure that exposes its underlying structure. The hierarchy infers the lexical structure in the sequence, so that it aids comprehension of the structure of input text and capture meaningful regularity. As a grammatical rule is a representation of a repeated phrase, a noun-phrase can be identified based on certain noun-phrase definition. Furthermore, by keeping track of the frequency of the grammatical rule replacing the phrases, we can capture the meaningful regular noun-phrases.

3. Extracting noun-phrases

In this section, we first describe the method of extracting noun phrases. Then we present the experiment and results that evaluate its performance with MEDLINE abstracts (https://www.ncbi.nlm.nih.gov/pubmed/).

3.1 Extracting noun-phrases method

The noun-phrase extraction schema includes 3 steps. The first step is to define noun-phrase patterns. The second step is the pre-processing of the input documents. The last step is to construct context-free grammar using the Sequitur algorithm and extract noun-phrases from the grammar rules.

3.1.1 Noun-phrase Pattern

We consider a noun-phrase as a group of words containing a noun that functions together as a noun. There are various definitions of noun phrases, some are simple and some are complex. In our work, we only identify the simple noun-phrases. A simple noun-phrase is a noun-phrase without relative clauses, and its head is the rightmost element and thus it has no right modification [14]. Many noun-phrases identification systems only identify simple noun-phrases [12, 14]. Noun-phrases are identified using a finite set of rules shown as following:

NP -> restOfNP | restOfNP conjunction restOfNP
restOfNP -> ADJ restOfNP | noun

ADJ -> adjective conjunction adjective | adjective

A noun-phrase can be a sequence of adjectives followed by a noun, or two noun-phrases combined with conjunction. And the adjective clause can be multiple adjectives either with or without conjunction. The words we are interested are adjective, noun and conjunctions. Therefore, before the steps of extracting noun-phrase, Part-of-speech (POS) tagging is applied to the input documents to label each word.

3.1.2 Pre-processing

This step consists of part-of-speech tagging, stemming and indexing. The Noun-phrase pattern is defined as a finite set of rules which are composed of noun, adjective and conjunction words. We first tag each word in input documents using part-of-speed tagging [15]. Because only noun, adjective, and conjunction are of interest, we eliminate those words that are not tagged as noun, adjective, and conjunction by replace these words with -1. Before indexing the words, we stem the words so that words having the same root (e.g., activate, activates, and activating) are collapsed to the same word for indexing. The result of the pre-processing is a sequence of number where each number is either -1 or representing a word of noun, adjective or conjunction.

3.1.3 Identify noun-phrases using the Sequitur algorithm

Using the Sequitur algorithm, the result of the pre-processing is re-represented as a set of context free grammatical rules. Each grammatical rule may or may not contain a noun-phrase or a partial noun-phrase. We go through each of the grammatical rule to extract noun-phrases from the rules. The algorithm for extracting noun-phrase is listed in Table 3.

Table 3. The noun-phrase extraction algorithm.

```
1 Extract_Noun_Phrase( ){
2  for each grammatical rule R, do
3    initial phrase as a empty string;

4    Extract_NP_from_Rule( R, phrase );
5  endfor;
6 }

7 Extract_NP_from_Rule( R, phrase){
8  for each symbol s in R, do
9     if s is a non-terminal symbol,
10       Extract_NP_from_Rule ( rule of s, phrase);
11    endif
12    elseif the word value of s is -1      //-1 is the delimiter of phrases
13       collect phrase to a phrase list;
14       reset phrase to a empty string;
15    endelseif
16    else
17       append the word value of s to the end of phrase;, and ensure it follows the noun-phrase pattern;
18    endelse;
19  endfor;
```

Given a context free grammatical rule, the noun-phrase extraction algorithm is designed to extract noun-phrases which are delimited by -1 within the rule. In the algorithm (Table 3), the "for" loop (lines 8-19) travels the right hand side of the rule and processes each symbol of the rule from left to right. If the symbol is a non-terminal symbol which refers to a grammatical rule, it will recursively call the noun-phrase extraction method on this rule. The variable **phrase** is functioned as a buffer that contains a partial phrase. If the symbol is a terminal symbol which refers to a word, the word will be appended to the end of **phrase**. At the same time, the algorithm also checks whether or not such appending action match as the noun-phrase pattern defined in the finite state rules (Section 3.1.1). The phrase will be sent to a collection of a phrase list when the value of the terminal symbol is -1. Because the phrase we refer in this study must contain more than one word, at this point, we can set the minimum and maximum numbers of words of a noun phrase, and only collect phrases with length within the range. Different rules may contain the same phrases. The duplicate phrases can be prevented using a hash table. We can also apply additional rules to prone noun-phrases, e.g. phrases with at least one non-stopped word.

3.2 Experiment

To evaluate our noun-phrase extraction method, we conducted an experiment that compares our method with other two methods. One is Phrase Parser from the set of SPECIALIST NLP tools developed by the Lexical Systems group of the Lister Hill National Center for Biomedical Communications (http://specialist.nlm.nih.gov/). In our work, we call this tool the NLP Phrase Parser. This NLP Phrase Parser tools is also used in the paper [12] for the purpose of comparison. This parser is primarily a barrier category parser, relying on parts of speech that have been already assigned to determine the beginnings and endings of phrases. This parser identifies the several kinds of phrases including noun-phrases, prep-phrases, verb-phrases, et. al. We are only interested in noun-phrases and prep-phrases. Because the prep-phrase is a noun-phrase with one or more prepositions ahead of it. We extract the noun-phrase within the prep-phrase when measuring the performance of NLP Phrase Parser.

The other method we compared with is MontyLingua [15]. MontyLingua is an end-to-end natural language understands for English. It can extract subject/verb/object tuples, extracts adjectives,

noun phrases and verb phrases, and other semantic information.

Because it is time consuming to identify noun-phrases manually, we use a small set of text containing 100 documents. These 100 documents are randomly selected from 3193 documents about soft-tissue sarcomas (STS) which are downloaded from MEDLINE. Title and abstract fields are kept for noun-phrase extraction. All these 100 documents are processed by our method, NLP Phrase parser and MontyLingua method, total numbers of noun-phrases identified by these three methods are 1958, 2401, and 2262, respectively. The Venn diagram of the result is shown in Figure 2.

In this experiment, in order to obtain the precision and the recall rate, all the simple noun-phrases are manually identified by a medical professional in advance. Here the precision value is defined as:

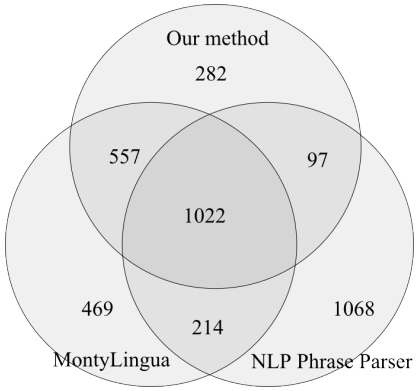

Figure 2. The comparison of extracting noun-phrase results

$$\text{precision} = \frac{\text{the number of noun phrases correctly identified by system}}{\text{the total number of noun phrases identified by system}}$$

The recall value is defined as:

$$\text{recall} = \frac{\text{the number of noun phrases correctly identified by system}}{\text{the total number of noun phrases in the documents}}$$

From the 100 documents, the medical expert identified 2156 simple noun-phrases that match the noun-phrase pattern defined in Section 3.1.1. We use these 2156 noun-phrases as the noun-phrases in the documents. And compare the results of the three methods with these 2156 noun-phrases to find out the noun-phrases that are correctly identified by system. There are some system-identified phrases that are correct noun-phrases, but not exactly match the human-identified noun-phrases. We consider this kind of phrases as correct noun-phrases. For example, we consider both phrases "brca1 and brca2 mutations" and "brca2 mutations" as correct noun-phrases.

Table 4 shows the precision and recall results for all three noun-phrase extraction methods. The results listed in Table 4 show that our noun-phrase extractor performed better than the NLP Phrase Parser and MontyLingua method. Our results show that the NLP Phrase Parser and MontyLingua method performed poor in identify the noun-phrases. However, it does not mean that these two methods are poor noun-phrase identifier. It could be because that their definitions of the noun-phrase pattern are different from ours.

Table 4. Comparison of the precision and recall values

Methods	Number of noun-phrases extracted by the noun-phrase extractor	Number of noun-phrases correctly identified by noun-phrase extractor	Precision (%)	Recall (%)
Medical expert	2156	N/A	N/A	N/A
NLP Phrase Parser	2401	1697	70.68	78.71
MontyLingua	2262	1626	71.88	75.42
Our method	1958	1821	93.00	84.46\

4. Key Noun-phrase Extraction

In the previous section, we described our noun-phrase extraction that extracts all the noun phrases in the documents. In this section, we present an efficient key noun-phrase extractor which is performed at the same time of noun-phrase extraction. We first describe the method that extracts key noun-phrases, and then present its performance evaluation.

4.1 Extracting key noun-phrase method

The Sequitur algorithm generates a set of grammatical rules that re-represent a sequence as a hierarchical structure which reveals its underlying structure. As a grammatical rule is a representation of a repeated subsequence, we can identify the frequently appeared subsequences by keeping track of the frequency of the grammatical rules replacing the subsequences. We can find the key noun-phrases by extracting the noun-phrases from these frequently appeared subsequences. Our key noun-phrase extraction method is an efficient method because we only need to extract the noun-phrases from the top frequent rules instead of extracting all the noun-phrases and then rank them.

The frequency of a rule is the number of times the subsequences represented by this rule appeared in the original input stream. Please notice that the frequency of a rule in this work is not the number of times the rule is used in the grammar. Computation of the frequency can be performed at the same time as the Sequitur algorithm constructs the context free grammar. Whenever a rule is used to replace a subsequence, its frequency is increased by one. With the grammatical rules sorted by their frequencies, key noun-phrases are the noun-phrases extracted from a certain number of top frequent rules. The key noun-phrase extraction algorithm can be easily modified from the noun-phrase extraction algorithm. The algorithm for extracting the key noun-phrase is listed in Table 5.

Table 5 The key noun-phrase extraction algorithm.

```
1 Extract_Key_Noun_Phrase( ){
2  sort_rule( );
2  for each grammatical rule R, and the frequency of R is larger than a threshold, do
3    initial phrase as a empty string;
4    Extract_NP_from_Rule( R, phrase );
5  endfor;
 }
```

As shown in Table 5, our key noun-phrase extraction algorithm is a simple algorithm that slightly modified from our noun-phrase extraction method. The key noun-phrase extracting algorithm first sorts the rules by the frequency in decreasing order, and then performs the noun-phrase extraction. To limit the number of key phrases extracted, we can either set a threshold or define a specific number of key phrases to be extracted. Only those rules with frequency higher than the threshold will be processed and the phrases will be extracted only from these rules in decreasing frequency order. Or the key noun-phrase extraction will be terminated once the number of extracted noun-phrase reaches the predefined number. The frequency of a phrase is the frequency of the rules from which it is extracted. For example, in Table 1, the context-free grammar for sequence of "abcbabcbabcabc" is:

$$S \rightarrow XXYY$$
$$X \rightarrow Yb$$
$$Y \rightarrow abc$$

Rule *Y* represents sequence of "*abc*". The frequency of rule *Y* is 4, assume subsequence "*bc*" is a noun-phrase, the frequency of the noun-phrase "*bc*" is also 4. Furthermore, if there is a noun-phrase, $\mathbf{NP_1}$, which is a subsequence of a noun-phrase, $\mathbf{NP_2}$, $\mathbf{NP_1}$ will be extracted before $\mathbf{NP_2}$. And the frequency of $\mathbf{NP_1}$ is not less than that of $\mathbf{NP_2}$. For example, in Table 1, rule *X* represents subsequence "*abcb*" with frequency value of 2. Assume "*abcb*" is also a noun-phrase, noun-phrase "*bc*" will be extracted before noun-phrase "*abcb*" because rule *Y* has higher frequency value than rule *X*. The frequency value of "*bc*" is 4, which is larger than that of "abcb".

From the algorithm shown in Table 5, we can conclude that our key noun-phrase extraction method is an efficient and effective method due to following 3 reasons.

1. **Noun-phrases are extracted from a certain number of top frequent rules.** Because the goal of this work is to extract the key noun-phrases instead of extracting all the noun-phrases, it is not necessary to process all the grammatical rules. By setting a threshold on the frequency of

the rules or define the number of key noun-phrases, we can dramatically reduce the number of rules that need to be processed.

2. **The rules with high frequency are usually rules representing short sequences. Processing time of these rules could be short.** Because the rule representing a long sequence, (R1), is very possible contains a rule representing a short sequence, (R2), which is either appear in rule R1 more than once or appear in other rules. For example, in Figure 1, rule Y is a non-terminal symbol of rule X, while rule Y is also a non-terminal symbol of rule S. The frequency of rule Y is larger than that of rule X. Therefore, rule Y is processed before X.

3. **Our method precisely computes the frequency of a noun-phrase which is also a sub-sequence of another noun-phrase.** A noun-phrase could be a sub-sequence of another noun-phrase. Normal existing key phrase extraction systems ignore the existence of the sub noun-phrase. Because our method computes the number of times a subsequence appeared in the original input text, the frequency of this kind of noun-phrases can be precisely obtain. For example, in the full text with the title of "Correspondence analysis of microarray time-course data in case-control design", which is publish in Journal of Biomedical Informatics, Volume 37, Issue 5 (October 2004), Pages: 358-365, the phrase "time-course" is also sub-phrase of "time-course experiment", "time-course pattern", "time-course data", "microarray time-course data", etc.. The total frequency number of "time-course" is 64. However, the frequency numbers of "time-course" computed using NLP and MontyLingua methods are only 4 because they didn't count the frequency of "time-course" when it is a sub-phrase of another noun-phrase. This means that our method in favor of those phrase with short length. As we know that most key phrases provided in the paper normally have length of 2 to 3, our method can produce a good performance in extracting key noun-phrase.

4.2 Experiment

We evaluate our key noun-phrase extracting method by measuring how well the system-generated key phrases match the author-provided key phrases for documents. To evaluate the effectiveness of our method, the precision, recall and F value were computed using the author-provided key phrases for documents. In this experiment, the precision value is defined as:

$$\text{precision} = \frac{\text{the number of key noun phrases that have been correctly extracted}}{\text{the value of ranking cutoff}}$$

The recall value is defined as:

$$\text{recall} = \frac{\text{the number of key noun phrases that have been correctly extracted}}{\text{the number of key phrases assiged by author}}$$

There is usually a trade-off between recall and precision, and either of them alone does not explain the system performance well. Therefore, the *F* measure was invented to show the combined results. The formula for *F* is:

$$F = 2 \times precision \times recall / (precision + recall).$$

We use 110 medical full text papers as the test documents in this evaluation. These papers are randomly selected from Journal of Biomedical Informatics published between 2005 and 2008. All these 110 papers have author assigned key phrases. The average length of these full text papers is 12.73, and the average number of author-assigned key phrases is 4.56. In this work, we consider the phrase length should be larger than 1. Therefore, we ignore those author-assigned key phrases with only one word. The average length of the author-assigned key phrases is 2.38. This number again shows that most key phrases provided in the paper are normally has length of 2 to 3.

We also want to know if our method is better than other method, so we compare our method with NLP and MontyLingua methods. We extract all the noun-phrases using these two methods respectively, and then count the frequencies of each phrase and rank them by frequency.

We calculate the average precision, recall and F value for all three methods on 110 papers when the number of extracted key phrases was 5, 10, 15, 20, 30, 40, 60, 80 and 100 respectively. The precision result is shown in Figure 3, the recall result is shown in Figure 4, and the F value result is shown in Figure 5. From these figures, we can see that our method performed better than NLP Phrase Parser and MontyLingua method at all the comparison points.

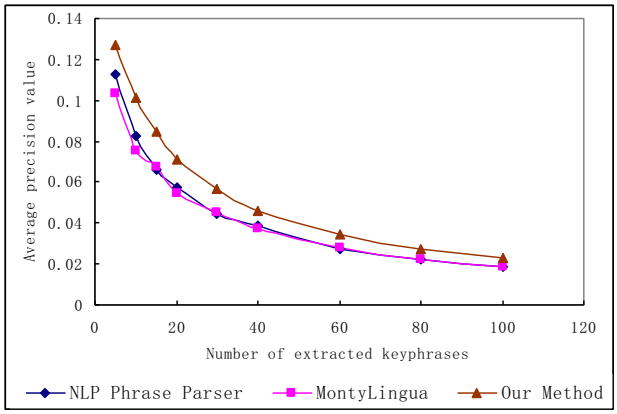

Figure 3. Comparison of the precision of our method, NLP Phrase Parser and MontyLingua method

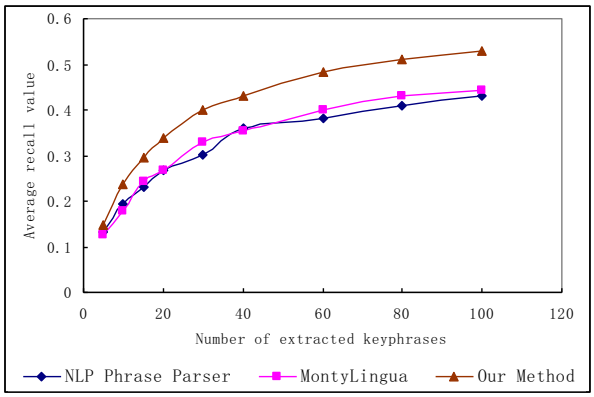

Figure 4. Comparison of the recall of our method, NLP Phrase Parser and MontyLingua method

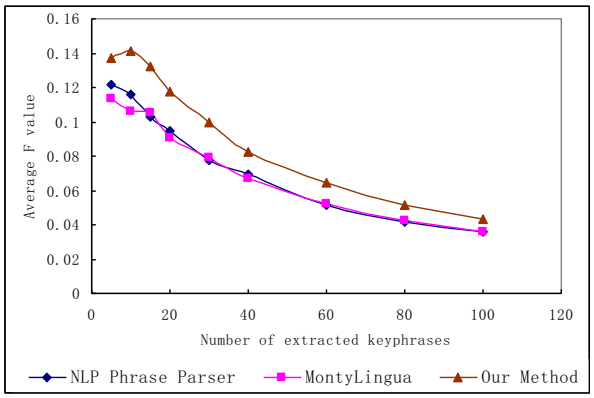

Figure 5. Comparison of the F value of our method, NLP Phrase Parser and MontyLingua method

However, these performance numbers are misleadingly low due to several reasons. One of them is that the author-assigned key phrases are usually only a small subset of the set of good quality key phrases for a given document. And some author-assigned key phrases are not simple noun-phrases. Another reason is that some author-assigned key phrases may not appear anywhere in the document they are assigned to. According to Turney [10], about 70% to 80% of author provided key phrases appear somewhere in the body of their documents, which means the highest possible average recall for a system could only be as high as 70%, even when all the phrases are extracted from the documents. A more accurate picture can be obtained by asking professional readers to rate the quality of the system output, which is a costly process.

5. Conclusion

This chapter presents a new topic extraction schema to identify the key noun-phrases by constructing context free grammar from input documents using the Sequitur algorithm. Noun-phrases are extracted from the grammatical rules. Key noun-phrases are identified from top frequency rules without extracting all the grammatical rules. Our key noun-phrase extraction method is effective in identifying key concepts from documents. The experimental results show that our method performed better than the systems it was compared to.

6. References

[1] S. Jonse, M. Mahoui, Hierarchical document clustering using automatically extracted keyphrase. In Proceedings of the 3rd International Asian conference on Digital Libraries. Seoul, Korea. 2000;113–20.

[2] Y. Zhang, N. Zincir-Heywood, E. Milios, Term-Based Clustering and Summarization of Web Page Collections. In Advances in Artificial Intelligence, Proceedings of the Seventeenth Conference of the Canadian Society for Computational Studies of Intelligence, London, ON, Canada, May 17–19, 2004;60-74.

[3] B.Kosovac, D.J. Vanier, T.M. Froese, Use of keyphrase extraction software for creation of an AEC/FM thesaurus. Electronic Journal of Information Technology in Construction, 2000;5:25–36.

[4] I. Witten, Browsing around a digital library. In Proceedings of the Australasian Computer Science Conference, 1999;1-14.

[5] E. Milios, Y. Zhang, B. He, L. Dong, Automatic Term Extraction and Document Similarity in Special Text Corpora. In Proceedings of the Sixth Conference of the Pacific Association for Computational Linguistics, Halifax, NS, Canada, August 22–25, 2003;275–284.

[6] Q. Li, Y.B. Wu, R.S. Bot, X. Chen, Incorporating document keyphrases in search results. In Proceedings of the 10th Americas Conference on Information Systems, New York, New York, 2004.

[7] S. Jones, M. Staveley. Phrasier: A system for

interactive document retrieval using keyphrases. In Proceedings of SIGIR'99, Berkeley, CA, 1999.

[8] B. Krulwich, C. Burkey, Learning user information interests through the extraction of semantically significant phrases, In M. Hearst and H. Hirsh, editors, AAAI Spring Symposium on Machine Learning in Information Access, 1996.

[9] Y.B. Wu , Q. Li , R.S. Bot, X. Chen, Domain-specific keyphrase extraction, Proceedings of the 14th ACM international conference on Information and knowledge management, Bremen, Germany, 2005.

[10] P. Turney, Learning to extract key phrases from text, Technical Report ERB-1057, National Research Council, Institute for Information Technology, 1999.

[11] F. Eibe, P.W. Gordon, W.H. Ian, G. Carl, N.G. Craig, Domain-Specific Keyphrase Extraction, Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 1999;668-673.

[12] Q. Li, Y.B.Wu, Identifying important concepts from medical documents. Journal of Biomedical Informatics, 2006;39:668-679.

[13] C.G. Nevill-Manning, I.H. Witten, Identifying Hierarchical Structure in Sequences: A linear-time algorithm. Journal of Artificial Intelligence Research, 1997;7:67-82.

[14] A.R. Aronson, T.C. Rindflesch, A.C. Browne, Exploiting a large thesaurus for information retrieval. In Proceedings of RIAO 1994:197-216.

[15] Liu, Hugo, MontyLingua: An end-to-end natural language processor with common sense, 2004. Available at: web.media.mit.edu/~hugo/montylingua.